I hereby certify that this correspondence is being filed via EFS-Web with the United States Patent and Trademark Office on March 22, 2010.

TOWNSEND and TOWNSEND and CREW LLP

By: / J. Matthew Zigmant /
J. Matthew Zigmant

PATENT
Attorney Docket No. 15114H-071400US
Client Ref. No. A00985

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

DHANOA, Kulwinder

Application No.: 10/749,910

Filed: December 30, 2003

For: SDRAM CONTROLLER

Confirmation No. 1395

Examiner: LEE, Chun Kuan

Technology Center/Art Unit: 2181

APPELLANTS' BRIEF UNDER 37 CFR §41.37

Mail Stop Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Commissioner:

Appellants submit this appeal brief in response to the Notice of Panel Decision from Pre-Appeal Brief Review mailed January 21, 2010.

1. REAL PARTY IN INTEREST

The real party in interest of the subject patent application is Altera Corporation, the assignee of the present application.

2. RELATED APPEALS AND INTERFERENCES

A Pre-Appeal Brief was filed for this application on December 11, 2009.

3. STATUS OF CLAIMS

Claims 1-2, 5-8, 11-13, and 18-21 are rejected.

4. STATUS OF AMENDMENTS

No Amendments After Final have been filed.

5. SUMMARY OF CLAIMED SUBJECT MATTER

In the following summary, Appellants have provided exemplary references to sections of the specification and drawings supporting the subject matter defined in the claims as required by 37 C.F.R. § 41.37. The specification and drawings also include additional support for other exemplary embodiments encompassed by the claimed subject matter. Thus, these references are only intended to be illustrative and not restrictive.

Data can be stored in a memory in a wrapped manner. Traditionally, such data is read from a first memory location, then a second memory and possibly other memory locations, before data in the first memory location is read again. Reading data from the first memory location twice wastes memory bandwidth. Accordingly, embodiments of the present invention provide buffers for storing this data. Data is read from the memory and stored in buffers. Data can be read from sub-buffers of the data buffers as needed, thereby avoiding redundant data reads from the memory.

A. Independent claim 1

Claim 1 recites a memory controller. (See pending specification, page 2, lines 24-33.) The memory controller includes at least one bus interface, each bus interface being for connection to at least one respective device for receiving memory access requests (page 5, lines 6-9), a memory interface, for connection to a memory device over a memory bus (page 4, lines 20-21), a plurality of buffers in the memory interface (page 5, lines 24-31), each of the plurality of buffers sized to store a data burst for a memory access request (page 6, lines 5-11), each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers (page 8, line 30 to page 9, line 5), and control logic, for placing received memory access requests into a queue of memory access requests. (Page 5, lines 17-22.)

In response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer. (Page 8, line 30 to page 9, line 5)

For a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic (page 9, lines 8-12), the beginning and end data for the wrapping memory access request being stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer by the memory interface (page 8, line 30 to page 9, line 5), the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request. (Page 9, lines 8-12)

The control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer. (Page 9, lines 24-26.)

B. Independent claim 7

Claim 7 recites a method of retrieving data with a memory controller including at least one bus interface for connection to at least one respective device for receiving memory access requests (page 5, lines 6-9), a memory interface for connection to a memory device over a memory bus (page 4, lines 20-21), a plurality of buffers in the memory interface (page 5, lines 24-31), and control logic for placing received memory access requests into a queue of memory access requests. (Page 5, lines 17-22.) The method includes. in response to a received memory access request requiring multiple data bursts over the memory bus, assigning each of the multiple data bursts to a respective buffer in the plurality of buffers in the memory interface, each of the plurality of buffers being sized to store a data burst for the memory access request, each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers. (Page 8, line 30 to page 9, line 5)

Data from each of said multiple data bursts is stored in the respective buffer in the memory interface. (Page 8, line 30 to page 9, line 5)

For a wrapping memory access request, data required for a beginning and an end of the wrapping memory access request is assigned to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request. (Page 9, lines 8-12.) A value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data is received, and the pointer is used to return to the indicated first sub-buffer to retrieve the end data. (Page 9, lines 24-26.)

C. Independent claim 13

Claim 13 recites a programmable logic device. The programmable logic device includes a memory controller. The memory controller includes at least one bus interface, each bus interface being for connection to at least one respective device formed within the programmable logic device for receiving memory access requests (page 5, lines 6-9), a memory interface, for connection to an external memory device over a memory bus (page 4, lines 20-21), a plurality of buffers in the memory interface (page 5, lines 24-31), each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers (page 6, lines 5-11), and control logic, for placing received memory access requests into a queue of memory access requests. (Page 5, lines 17-22.)

In response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer. (Page 6, lines 5-11)

For a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the wrapping memory access request being stored concurrently from a single data burst in the respective sub-buffers by the memory interface, the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request. (Page 9, lines 8-12.)

The control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single buffer. (Page 9, lines 24-26.)

D. Independent claim 18

Claim 18 recites a memory controller. The memory controller includes at least one bus interface, each bus interface being for connection to at least one device for receiving memory access requests (page 5, lines 6-9), a memory interface, for connection to a memory device over a memory bus (page 4, lines 20-21), a plurality of buffers in the memory interface (page 5, lines 24-31), each of the plurality of buffers sized to store a data burst for a memory access request (page 6, lines 5-11), and control logic, for placing received memory access requests into a queue of memory access requests. (Page 5, lines 17-22.)

For a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to sub-buffers of a single buffer by the control logic. (Page 8, line 30 to page 9, line 5)

The control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single buffer. (Page 9, lines 24-26.)

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to be reviewed on appeal are:

Whether claims 1-2, 7-8, 13, and 18-19 are unpatentable under 35 U.S.C. 103(a) as being obvious in view of Gray et al., United States patent number 6,816,923, in view of Abramson et al., United States patent number 6,499,077, Iizuka et al., United States patent number 5,581,530, and Nguyen et al., United States patent number 5,355,326, whether claims 5, 11, and 20 are unpatentable under 35 U.S.C. 103(a) as being obvious in view of Gray in view of Abramson, Iizuka, Nguyen and Kuronuma et al., United States patent number 6,859,848, and whether claims 6, 12, and 21 are unpatentable under 35 U.S.C. 103(a) in view of Gray in view of Abramson, Iizuka, Nguyen and "Microsoft Computer Dictionary."

7. ARGUMENT

For purposes of this appeal, claims 2, 5-8, 11-13, and 18-21 may stand or fall with respect to independent claim 1. No admissions are made by the grouping of claims. Appellants reserve the right to pursue any claims that are not specifically argued in the subsequent prosecution of the present application, and in any continuation application.

A. Rejection of claim 1 under 35 U.S.C. 103(a) as being unpatentable over Gray in view of Abramson, Iizuka, and Nguyen.

Claim 1 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Gray in view of Abramson, Iizuka, and Nguyen. But this combination of cited references does not show or suggest "a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request," in combination with the other recited limitations.

The pending office action cites several passages in Iizuka as showing or suggesting this feature, but appears to recognize that Iizuka does not show or suggest that the buffers are sized to store a data burst since Iizuka's buffers are sized to store a number of samples. (See pending office action, page 8, second full paragraph.) The pending office action further cites Gray as showing storing a data burst from memory. (See pending office action, page 2, last paragraph.) The rejection is thus understood to be that since Gray stores data bursts and Iizuka shows buffers sized to hold a number of samples, the combination results in buffers sized to hold data bursts.

The pending advisory action states that the motivation for combining Iizuka and Gray in this manner is "for the benefit of implementing a simplified structure and providing an optimal priority order for data transferring." (See pending advisory action, page 2, third paragraph.)

Iizuka shows that a simplified structure is desired in column 2, lines 26-29. In this passage, Iizuka shows that the simplified structure can be used to "freely edit recorded data without actually rewriting the data." (*id.*) Iizuka further shows that editing data requires cross-fade in order to prevent an unnatural sound. (See Iizuka, column 1, lines 52-59.) These cross-fades have a length of several seconds, and conventionally require data to be rewritten. (id.) Iizuka shows that this rewriting can be avoided by storing data in a cross-fade memory, and reading the data out when required. (See Iizuka, Figure 12 and column 23, lines 28-54.) Accordingly, the simplified structure shown by Iizuka relates to using a cross-fade memory; it does not relate to the sizing of buffers.

Iizuka shows that a priority order for data transferring can be determined in a number of ways, including "a degree of emergency." (See Iizuka, column 32, lines 14-19.) Iizuka does not show that the priority order is optimized by sizing buffers in any particular manner. Accordingly, there is no motivation to combine the references in the manner suggested to show or suggest that each of the plurality of buffers is sized to store a data burst for a memory access request.

The pending office action recognizes that Abramson does not show or suggest this feature. (See pending office action, page 10, second paragraph.) Nguyen adds nothing to this, and these references do not provide motivation for including this feature. Accordingly, the combination of cited references does not show or suggest a plurality of buffers in the memory interface, each of the plurality of buffers <u>sized to store a data burst for a memory access request</u>, in combination with the other elements recited by the claim.

Claim 1 further recites "wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic." The combination of cited references does not show or suggest this feature.

The pending office action cites Figure 8 of Iizuka as showing or suggesting this feature. (See pending office action, page 8, last paragraph.)

In Figure 8, Iizuka shows three buffers, 9-1 to 9-3. (See Iizuka, Figure 8.) Each buffer stores a sample. (See Iizuka, column 11, lines 20-25.) Each buffer is a FIFO operating as a ring buffer. (*id.*) In this configuration, a start and end address for each sample are stored in a buffer 9-1 to 9-3. Data in buffers 9-1 to 9-3 are transferred to a hard drive using DMA transfers. (See Iizuka, Figure 8.)

Thus, Iizuka does not show a wrapping memory access request. Instead, Iizuka shows that each sample is stored in a single buffer. Iizuka does not show that a sample needs to be wrapped using multiple buffers. Because of this, Iizuka does not show that data required for a beginning and end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer. Instead, Iizuka shows that the beginning and end of each sample is stored in its own buffer 9-1 to 9-3.

The advisory office action appears to recognize that Iizuka does not show this feature, and instead appears to indicate that Iizuka in combination with one or more other references shows this feature. (See pending advisory action, page 2, fifth paragraph.) The motivation for this combination is again "for the benefit of implementing a simplified structure and providing an optimal priority order for data transferring." (*id.*)

Again, Iizuka shows that a simplified structure is provided to avoid rewriting data during a cross-fade and that the priority order can be determined according to a degree of emergency or other criteria. Neither of these concerns suggests that data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer, and the cited references do not provide motivation for including this feature.

Accordingly, the combination of cited references does not show or suggest wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, in combination with the other elements recited by the claim.

Claim 1 further recites "wherein the control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that <u>the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer</u>." The combination of cited references does not show or suggest this feature.

The pending office action cites Nguyen as showing or suggesting this feature. (See pending office action, page 9, last paragraph.)

Nguyen shows the use of input and output pointers. (See Nguyen, Figure 2.) The input pointer points to the next slot involved in an input operation, while the output pointer points to the next slot involved in an output operation. (See Nguyen, column 6, lines 4-7.) Nguyen does not show that these pointers allow control logic <u>to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer</u>. Instead, Nguyen shows an input pointer that points to a next slot involved in an input operation and an output pointer that points to the next slot involved in an output operation.

The pending office action appears to recognize that Nguyen does not show this feature, and instead states that the previous office action "relied mainly on Iizuka for the teaching" of this feature. (See pending office action, page 4, second paragraph.) The pending office action further cites several passages in Iizuka. However, as described above, Iizuka does not show memory wrapping, and thus cannot show a pointer that allows control logic to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer.

Moreover, no motivation for combining Iizuka and Nguyen in a manner to show this feature is given in either the pending final office action or the pending advisory action.

The pending office action recognizes that Gray and Abramson do not show or suggest this feature. (See pending office action, page 12, first full paragraph.) Also, the cited references do not provide motivation for including this feature. Accordingly, the combination of cited references does not show or suggest wherein the control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer, in combination with the other elements recited by the claim.

For at least these reasons, claim 1 should be allowed.

8. CONCLUSION

For these reasons, it is respectfully submitted that the rejection should be reversed.

Respectfully submitted,

/ *J. Matthew Zigmant* /

J. Matthew Zigmant
Reg. No. 44,005

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 415-576-0200
Fax: 415-576-0300
62521243 v1

9. CLAIMS APPENDIX

Claim 1. A memory controller, comprising:

at least one bus interface, each bus interface being for connection to at least one respective device for receiving memory access requests;

a memory interface, for connection to a memory device over a memory bus;

a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers; and

control logic, for placing received memory access requests into a queue of memory access requests,

wherein, in response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer,

wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the wrapping memory access request being stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer by the memory interface, the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request, and

wherein the control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single respective buffer.

Claim 2. A memory controller as claimed in claim 1, wherein, when returning data to the respective device from which a memory access request requiring multiple data bursts over the memory bus was received, data is read out from a first part of the single respective buffer, then data is read out from at least one other of said buffers, then data is read out from a second part of the single respective buffer.

Claims 3-4. (Cancelled)

Claim 5. A memory controller as claimed in claim 1, wherein the control logic determines whether a received read access request is a wrapping request which requires multiple memory bursts, and, if so, the control logic allocates each of said memory bursts to a respective one of said buffers.

Claim 6. A memory controller as claimed in claim 1, wherein the memory controller is a SDRAM controller, and said memory interface is suitable for connection to a SDRAM memory device over said memory bus.

Claim 7. In a memory controller including at least one bus interface for connection to at least one respective device for receiving memory access requests, a memory interface for connection to a memory device over a memory bus, a plurality of buffers in the memory interface, and control logic for placing received memory access requests into a queue of memory access requests, a method of retrieving data comprising:

in response to a received memory access request requiring multiple data bursts over the memory bus, assigning each of the multiple data bursts to a respective buffer in the plurality of buffers in the memory interface, each of the plurality of buffers being sized to store a data burst for the memory access request, each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers;

storing data from each of said multiple data bursts in the respective buffer in the memory interface;

for a wrapping memory access request, assigning data required for a beginning and an end of the wrapping memory access request to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request;

recording a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data; and

using the pointer to return to the indicated first sub-buffer to retrieve the end data.

Claim 8. A method as claimed in claim 7, further comprising, when returning data to the respective device from which a memory access request requiring multiple data bursts over the memory bus was received, reading data out from a first part of the single respective buffer, then reading data out from at least one other of said buffers, then reading data out from a second part of the single respective buffer.

Claims 9-10. (Cancelled)

Claim 11. A method as claimed in claim 7, further comprising determining whether a received read access request is a wrapping request which requires multiple memory bursts, and, if so, performing the step of assigning each of said memory bursts to a respective one of said buffers.

Claim 12. A method as claimed in claim 7, wherein the memory controller is a SDRAM controller, and said memory interface receives data from a SDRAM memory device over said memory bus in SDRAM bursts.

Claim 13. A programmable logic device, wherein the programmable logic device includes a memory controller, comprising:

at least one bus interface, each bus interface being for connection to at least one respective device formed within the programmable logic device for receiving memory access requests;

a memory interface, for connection to an external memory device over a memory bus;

a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers, each sized to store a data beat of the data burst stored in one of the corresponding plurality of buffers; and

control logic, for placing received memory access requests into a queue of memory access requests,

wherein, in response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer,

wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the wrapping memory access request being stored concurrently from a single data burst in the respective sub-buffers by the memory interface, the storing of the beginning and end data in the single respective buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory request being cached in one or more of the respective sub-buffers until needed for transfer in response to the wrapping memory access request; and

wherein the control logic records a value of a pointer indicating a first sub-buffer of the single respective buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single buffer.

Claims 14-17. (Cancelled)

Claim 18. A memory controller, comprising:

at least one bus interface, each bus interface being for connection to at least one device for receiving memory access requests;

a memory interface, for connection to a memory device over a memory bus;

a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request; and

control logic, for placing received memory access requests into a queue of memory access requests,

wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to sub-buffers of a single buffer by the control logic, and

wherein the control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated first sub-buffer to retrieve the end data from the single buffer.

Claim 19. A memory controller as claimed in claim 18, wherein, when returning data to the device from which a memory access request requiring multiple data bursts over the memory bus is received, data is read out from a first part of the single buffer, then data is read out from at least one other of the buffers, then data is read out from a second part of the single buffer.

Claim 20. A memory controller as claimed in claim 18, wherein the control logic determines whether a received read access request is a wrapping request which requires multiple memory bursts, and, if so, the control logic allocates each of the memory bursts to one of the buffers.

Claim 21. A memory controller as claimed in claim 18, wherein the memory controller is a SDRAM controller, and the memory interface is suitable for connection to a SDRAM memory device over the memory bus.

10. EVIDENCE APPENDIX

None.

11. RELATED PROCEEDINGS APPENDIX

None.